HP200315774
(ip504-0478)
Ref. 1.

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 09-231069
(43)Date of publication of application : 05.09.1997

(51)Int.Cl. G06F 9/24
G06F 9/445

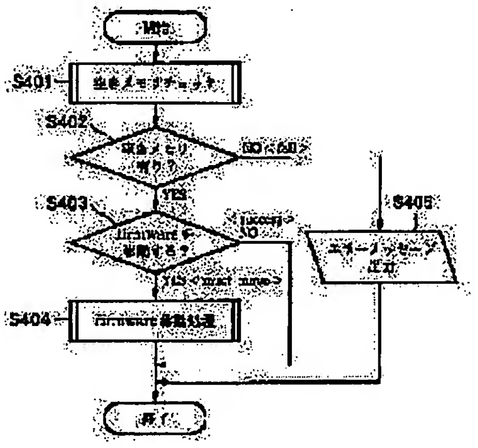
(21)Application number : 08-034927
(22)Date of filing : 22.02.1996

(71)Applicant : CANON INC
(72)Inventor : NOZAKI JUN

(54) METHOD AND DEVICE FOR INFORMATION PROCESSING

(57)Abstract:

PROBLEM TO BE SOLVED: To minimize the address dependent parts requested by the firmware by adding a flexible memory management mechanism to the firmware, to reduce the packaging manhour of an OS (operating system), i.e., a client, and also to easily package more OSs.
SOLUTION: The firmware and an OS loader are stored in a RAM. When a request is received to secure a memory area to the RAM while an OS is loaded by the OS loader, the firmware checks an idle memory (S401). If a memory area having a requested size is available, a memory area is secured in response to the corresponding request (S403). If it is impossible to secure the requested memory area, an error is outputted (S405). Then the firmware is moved as long as a memory area is secured as requested by the movement of the firmware (S404). Thus the requested memory area is secured.



LEGAL STATUS
[Date of request for examination]
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

Ref. 1 (Claims)

CLAIMS

[Claim(s)]
[Claim 1] The information processor characterized by having a 1st load means to load the firmware corresponding to the hardware of an information processor to memory, a 2nd load means to use hardware resources through said firmware and to load an operating system to said memory, and a secured means to secure the memory area of the size demanded by said 2nd load means and said operating system on said memory.
[Claim 2] Said secured means is an information processor according to claim 1 characterized by securing the memory area of size specified by said 2nd load means and said operating system from the address with which said memory was specified.
[Claim 3] It is the information processor according to claim 1 or 2 which is further equipped with the table showing the busy condition of the room in said memory, and is characterized by said secured means securing the memory area demanded with reference to said table.
[Claim 4] Said table is an information processor according to claim 3 characterized by what is memorized by nonvolatile memory.
[Claim 5] The information processor according to claim 3 characterized by having further an updating means to update said table based on the secured condition of the memory area by said secured means.
[Claim 6] The information processor according to claim 3 characterized by having further an edit means to edit said table.
[Claim 7] The information processor according to claim 1 characterized by having further a migration means to move in the storing field of the firmware concerned if needed, on the occasion of reservation of the memory area by said secured means.
[Claim 8] Said migration means is an information processor according to claim 7 characterized by being one of the functions with which said firmware is equipped.
[Claim 9] It is the information processor according to claim 7 which it has further the table showing the busy condition of the room in said memory, and said secured means secures the memory area demanded with reference to said table, and is characterized by said migration means determining the migration place of said firmware with reference to said table.
[Claim 10] The information processor according to claim 9 characterized by having further an updating means to update said table based on modification of the memory busy condition by said secured means and the migration means.
[Claim 11] The start address which shows the head storing location of a program at least about said firmware, The program control table which registers the return address after this firmware migration by said migration means with the relative address in this firmware, Based on the migration result of said firmware by said migration means, said program start address of said program control table is changed into the address of a migration place. The information processor according to claim 7 characterized by having further a continuation means to continue activation of this firmware from the location shown by said return address.
[Claim 12] The information-processing approach characterized by to have the 1st load process which is the information-processing approach at the time of starting of an information processor, and loads the firmware corresponding to the hardware of said information processor to memory,

the 2nd load process which uses hardware resources through said firmware and loads an operating system to said memory, and the secured process which secure the memory area of the size demanded by said 2nd load process and said operating system on said memory.
[Claim 13] Said secured process is the information processing approach according to claim 12 characterized by securing the memory area of size specified by said 2nd load process and said operating system from the address with which said memory was specified.
[Claim 14] It is the information processing approach according to claim 12 or 13 which is further equipped with the table showing the busy condition of the room in said memory, and is characterized by said secured process securing the memory area demanded with reference to said table.
[Claim 15] Said table is the information processing approach according to claim 14 characterized by what is memorized by nonvolatile memory.
[Claim 16] The information processing approach according to claim 14 characterized by having further the updating process which updates said table based on the secured condition of the memory area by said secured process.
[Claim 17] The information processing approach according to claim 14 characterized by having further the edit process which edits said table.
[Claim 18] The information processing approach according to claim 12 characterized by having further the migration process which moves in the storing field of the firmware concerned if needed on the occasion of reservation of the memory area by said secured process.
[Claim 19] Said migration process is the information processing approach according to claim 18 characterized by being one of the functions with which said firmware is equipped.
[Claim 20] It is the information processing approach according to claim 18 which it has further the table showing the busy condition of the room in said memory, and said secured process secures the memory area demanded with reference to said table, and is characterized by said migration process determining the migration place of said firmware with reference to said table.
[Claim 21] The information processing approach according to claim 20 characterized by having further the updating process which updates said table based on modification of the memory busy condition by said secured process and the migration process.
[Claim 22] The start address which shows the head storing location of a program at least about said firmware, It has the program control table which registers the return address after this firmware migration by said migration process with the relative address in this firmware. Based on the migration result of said firmware by said migration process, said program start address of said program control table is changed into the address of a migration place. The information processing approach according to claim 18 characterized by having further the continuation process which continues activation of this firmware from the location shown by said return address.
[Claim 23] It is the computer-readable memory in which the program code of the information processing approach at the time of starting of an information processor was stored. The code of the 1st load process which loads the firmware corresponding to the hardware of said information processor to memory, The code of the 2nd load process which uses hardware resources through said firmware and loads an operating system to said memory, Computer-readable memory characterized by having the code of the secured process which secures the memory area of the size demanded by said 2nd load process and said operating system on said memory.

[Translation done.]

Ref. 1 (Detailed Description)

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[Field of the Invention] This invention relates to the especially suitable firmware for a personal computer, a workstation, etc. about an information processor and the information processing approach.
[0002]
[Description of the Prior Art] Conventionally, in the personal computer and the workstation, the firmware is the design which was conscious of the specific operating system in many cases. For the reason, the firmware itself serves as a design depending on the specific address, and the memory control unit which a firmware offers had only the very simple thing.
[0003]
[Problem(s) to be Solved by the Invention] In recent years, the attempt which performs various operating systems is made on the same platform, and, for that purpose, the standardization of a firmware and flexible-izing of structure which a platform mounts are called for. However, many of conventional firmwares have that the customize activity according to a user in that the active position on the memory is immobilization **** is [much] required for beforehand. When it mounts the operating system newly developed or transplanted for the reason, the man day for fitting an operating system to the room which the form wear defines, and the address which a firmware uses occurs. When transplanting an operating system with the kernel depending on a specific firmware, or OS loader to a platform with another firmware, this man day is generated especially notably.
[0004] By making this invention in view of the above-mentioned problem, and preparing a flexible memory control unit in a firmware, minimize the address dependence section which a firmware requires, the mounting man day of the operating system which is a client is made to mitigate, and it aims at offering the information processing approach and equipment which enable mounting of more operating systems easily.
[0005]
[Means for Solving the Problem] The information processor of this invention for attaining the above-mentioned purpose is equipped with the following configurations. That is, it has a 1st load means to load the firmware corresponding to the hardware of an information processor to memory, a 2nd load means to use hardware resources through said firmware and to load an operating system to said memory, and a secured means to secure the memory area of the size demanded by said 2nd load means and said operating system on said memory.
[0006] Moreover, the information processing approach of this invention for attaining the above-mentioned purpose is equipped with the following configurations. That is, it is the information processing approach at the time of starting of an information processor, and has the 1st load process which loads the firmware corresponding to the hardware of said information processor to memory, the 2nd load process which uses hardware resources through said firmware and loads an operating system to said memory, and the secured process which secures the memory area of the size demanded by said 2nd load process and said operating system on said memory.
[0007]

[Embodiment of the Invention] Hereafter, with reference to an attached drawing, 1 suitable operation gestalt of this invention is explained.
[0008] Drawing 1 is a block diagram showing the outline configuration of the information processor by this operation gestalt. In this drawing, 101 expresses arithmetic and program control, i.e., CPU. Moreover, for ROM (Read Only Memory) in which 102 stored the firmware, and 103, as for nonvolatile memory and 105, RAM (Random Access Memory) and 104 are [a magnetic disk drive and 106] system consoles.
[0009] Generally, in the case of an information processor with a configuration like drawing 1 , the process to loading of an operating system becomes the following procedures after a system startup (at the time of a power source ON).
[0010] (1) Loading of the expansion to expansion of the firmware program in jump and (2) ROM, starting and a diagnosis of the peripheral device according control to (3) firmwares to the specific address after a diagnosis of hardware, and initialization, and in ROM, initialization, read-out from the magnetic disk drive of OS loader by (4) firmwares, and RAM, the kernel by (5) OS loader, and a resource. At this time, the I/O demand to a firmware from OS loader etc. occurs, and a firmware shifts control to OS which completed (6) loads which answer this I/O demand.
[0011] Below, based on the above-mentioned process, the load process of the operating system at the time of the system startup in this operation gestalt is explained. Drawing 5 is a flow chart showing the control procedure at the time of the system startup in this operation gestalt.
[0012] (1) Perform diagnosis of hardware, and initialization and jump to the specific address (step S301). Processing of this operation gestalt performs actuation generally known about this processing. Therefore, detailed explanation is omitted here.
[0013] (2) Develop and start the firmware program in ROM to RAM (step S302).
[0014] In this operation gestalt, the firmware program has structure as shown in drawing 2 . In drawing 2 , 201 is a header and it is shown that the following data are firmware programs. 202 is a firmware coding region and the code of the firmware program containing the initialization routine mentioned later is stored. Moreover, 203 is a firmware data area and is a data area for a firmware program to use. In addition, the firmware program by this operation gestalt serves as a relocatable activation module without an address dependency. And the conversion to an effective address (effective address) has structure dynamically searched for within a program by the formula of =(effective address) (program internal phase pair address)+ (program start address) if needed.
[0015] Now, in case the above-mentioned firmware program is transmitted to RAM103 from ROM102 by the initialization routine within a firmware program, the memory management table shown in drawing 3 is read from NVRAM104. In drawing 3 , 301 expresses the number of entries in this memory management table. 302 expresses the attribute of the memory area pinpointed by 303,304. There are the following as an attribute.
[0016] - Free - Handler whose field whose specified field is a free area, and of which - Firmware:assignment was done is a firmware field: The specified field is - Not which are system areas, such as an exception interrupt handler. The field accessable(d) : specified is - Load which is access needlessness. The field address(ed) : specified is - Client which is the loading field of OS loader (bootstrap program). The field area(ed) : specified is the field of clients other than OS loader.
[0017] A starting address 303 shows the start address of the memory area classified for every above attributes, and size 304 expresses the size of the field.
[0018] Now, initialization routine investigates the above-mentioned memory management table, and carries out loading of the firmware program from the "starting address" of an entry with which the attribute serves as Firmware. A default is adopted when a managed table is not found in NVRAM104. In addition, the memory management table in NVRAM104 can be edited by the user by the utility which a firmware program offers if needed.
[0019] If loading to RAM103 of a firmware program is completed, initialization routine will create the program control table shown in drawing 4 . In drawing 4 , 401 is a program start address and the start address of the present firmware program is stored. 402 is a data area start address and the head of the data area of the firmware program concerned is stored in the program internal

phase pair address. 403 is a return address and the address for the restart after the firmware program migration processing mentioned later is stored in the program internal phase pair address. 404 is a register shunting field and is a field which evacuates the contents of a register at the time of the firmware program migration processing mentioned later.
[0020] Initialization routine sets a value as the program start address 401 of a program control table, and the data area start address 402, and passes control to the firmware program in RAM.
[0021] (3) A firmware performs diagnosis of a peripheral device and initialization (step S303). In this operation gestalt, a diagnosis of the peripheral device by the firmware and initialization perform a general procedure, and omit explanation here.
[0022] (4) Expansion to read-out from the magnetic disk drive of OS loader by the firmware, and RAM, and starting (step S304)
A firmware program develops OS loader (bootstrap program) on RAM103 from a magnetic disk drive 105. The loading address at this time is Load of the memory management table mentioned above. It asks from an entry with an address attribute. After loading completion of OS loader moves control to the OS loader concerned.
[0023] (5) Answer the I/O demand generated from OS loader to a firmware at the time of the various resource expansions by OS loader (step S305).
[0024] OS loader reads various resources including the kernel (kernel) of OS from a magnetic disk drive 105, and develops them on RAM103. However, it cannot but depend for these I/O actuation (the output to console 106 grade is also included) on a firmware until the own device driver of OS is developed on RAM103 and an execution environment is ready. Therefore, a client (OS loader or operating system) and a firmware need to live together on memory till then. So, with this operation gestalt, the program interface of a memory requirement was prepared in the firmware program.
[0025] A client makes it unnecessary to ask a client for the memory use depending on a firmware at the same time it aims at prevention of mutual destruction of the memory between a firmware and a client by performing a memory requirement to a firmware through this program interface on the occasion of expansion of the various resources on RAM103.
[0026] Drawing 6 is a flow chart showing processing of the firmware at the time of receiving a memory requirement. In this drawing, step S401 is empty memory check processing, and asks for the free area of RAM103 with reference to the memory management table mentioned above here. When the specified free area is found, a memory management table is updated and it returns.
[0027] In addition, in the firmware program by the operation gestalt of this invention, about a memory requirement, two kinds of specification methods of the demand by the demand and starting address by - cutting tool length, and cutting tool length are mounted, and empty memory check processing is equivalent to these assignment.
[0028] When it is vacant by reference and the program interface of memory check processing is shown, it is int getFreeArea (starting address of which the IN:void*start_Address demand was done (when there is no demand of a starting address - 1)).
IN: int request_Memory_size Demanded memory size OUT:void*allocated_Address It becomes like the start address of the room allocated by the demand.
[0029] Above-mentioned empty memory check processing returns three kinds of return information as follows. Namely, (1) success:(2) by which the specified memory area was allocate (d) fail: The memory area where allocate of the specified memory area was specified by migration of an impossibility and a (3) must move:firmware is allocate(d).
[0030] When success is returned, it has the returned address value and returns to a client. Moreover, when fail is returned, a firmware carries out an error message output at a console, and carries out an error return at a client (steps S402 and S405). (about subsequent processing, it is dependent on a client) Furthermore, control is put into 404 firmware migration processing when must move is returned (steps S402, S403, and S404). In addition, in case control is put into firmware migration processing, the starting address of the entry of a Firmware attribute of the memory management table shown by drawing 2 is updated so that allocate of the specified memory area may be possible.

[0031] Drawing 7 is a flow chart showing the contents of processing of firmware migration processing (step S404). In step S501, the start address of the firmware program of a migration place is stored in the program start address 401 of the program control table mentioned above. The starting address of the entry of the Firmware attribute of the memory management table updated by empty memory check processing is used for this address.
[0032] Next, in step S502, it stores in the return address 403 of a program control table by using the return address of this processing as an activation pointer. This address turns into the address performed by the beginning, after passing through the reentry processing mentioned later after firmware program migration. For example, if the address of the location in the end of step S404 of drawing 6 (location in the end of step S505 of drawing 7) is set, after finishing the below-mentioned reentry processing, the firmware concerned will end processing of drawing 6 and will shift to the next processing succeedingly.
[0033] In step S503, the present register value is stored in the register save area 404 of a program control table. Then, in step S504, a firmware program is transmitted to a migration place. At step S505, it jump(s) to reentry processing of the firmware program of a migration place. It asks for the address of reentry processing from the sum of the program relative entry address of reentry processing, and the migration place address.
[0034] Drawing 8 is a flow chart showing actuation of the reentry processing in this operation gestalt. In step S601, the memory management table created by above-mentioned empty memory check processing is written in nonvolatile memory (NVRAM104). This is for aiming at mitigation of the overhead accompanying migration processing of a firmware by arranging a firmware to the suitable address in advance with reference to this at the time of next boot.
[0035] Next, in step S602, the register value stored by firmware migration processing mentioned above is restored. And in step S603, to the entry point stored as a return address 403 by firmware migration processing mentioned above, the program internal phase pair jump is performed and the firmware program concerned is resumed.
[0036] Migration of a firmware is performed as mentioned above and it becomes possible to secure appropriately the memory area demanded from OS loader. And termination of loading of the resource by OS loader moves control to the operating system concerned (step S306).
[0037] As explained above, according to this operation gestalt, the flexible design of OS loader (bootstrap program) which is a client is attained by having prepared the program interface for the memory area demand from a client in the firmware.
[0038] Moreover, according to the above-mentioned operation gestalt, mounting of a client with a strong memory dependency becomes easy.
[0039] Moreover, according to the above-mentioned operation gestalt, a deployment of memory is achieved in order for the firmware itself to rearrange within memory dynamically if needed. Moreover, according to the above-mentioned operation gestalt, a pointer is managed as a program relative address and the start address of a program is held. For this reason, when a program makes a note and the physical location inside changes, in case it is made to operate continuously, it becomes easy to fit various pointers with a program counter to the new address. That is, the reboot after migration of a program becomes easy.
[0040] Moreover, in the boot after it, in order to memorize the memory arrangement which the last boot took to nonvolatile memory according to the above-mentioned operation gestalt, in order to carry out suitable memory arrangement to boot processing of the last client, compaction of boot time is achieved.
[0041] Furthermore, by editing the memory management table of NVRAM104 after a system startup, since customize of memory arrangement is possible, more flexible memory management becomes possible.
[0042] In addition, although the firmware program was used as the continuous program module with the above-mentioned operation gestalt also including the data area which it manages, this is divided into suitable fragmentation and you may make it manage on memory. Since partial migration is attained on the occasion of migration of a firmware program in this case, it becomes rearrangeable [more flexible and high-speed memory management and a firmware].
[0043] Moreover, although the firmware in the information processor on condition of activation of

an operating system was mentioned and the actuation at the time of loading of a common operating system was stated with the above-mentioned operation gestalt, application of this invention is not restricted to this. For example, a client is able to be [a firmware] exchangeable for example, mount a firmware in a home video game special-purpose machine etc. Here, a client receives service from a firmware and means what is loaded on memory and performed. By usual computer, it is OS or its loader, and if it is a game special-purpose machine, the game program itself will be pointed out. Moreover, the condition that a client changes for every game like a game program [in / that a client is exchangeable / a game special-purpose machine] is said.
[0044] In addition, even if it applies this invention to the system which consists of two or more devices (for example, a host computer, an interface device, a reader, a printer, etc.), it may be applied to the equipments (for example, a copying machine, facsimile apparatus, etc.) which consist of one device.
[0045] Moreover, it cannot be overemphasized by the purpose of this invention supplying the storage which recorded the program code of the software which realizes the function of the operation gestalt mentioned above to a system or equipment, and carrying out read-out activation of the program code with which the computer (or CPU and MPU) of the system or equipment was stored in the storage that it is attained.
[0046] In this case, the function of the operation gestalt which the program code itself read from the storage mentioned above will be realized, and the storage which memorized that program code will constitute this invention.
[0047] As a storage for supplying a program code, a floppy disk, a hard disk, an optical disk, a magneto-optic disk, CD-ROM, CD-R, a magnetic tape, the memory card of a non-volatile, ROM, etc. can be used, for example.
[0048] Moreover, it cannot be overemphasized that it is contained also when the function of the operation gestalt which performed a part or all of processing that OS (operating system) which is working on a computer is actual, based on directions of the program code, and the function of the operation gestalt mentioned above by performing the program code which the computer read is not only realized, but was mentioned above by the processing is realized.
[0049] Furthermore, after the program code read from a storage is written in the memory with which the functional expansion unit connected to the functional add-in board inserted in the computer or a computer is equipped, it cannot be overemphasized that it is contained also when the function of the operation gestalt which performed a part or all of processing that CPU with which the functional add-in board and functional expansion unit are equipped based on directions of the program code is actual, and mentioned above by the processing is realized.
[0050] Although the program code corresponding to the flow chart explained previously will be stored in the storage when applying this invention to the above-mentioned storage, when it explains briefly, each module shown in the example of a memory map of drawing 9 will be stored in a storage.
[0051] Namely, what is necessary is just to store the program code of each module of the "1st load processing module", the "2nd load processing module", and a "secured processing module" in a storage at least.
[0052] Here, the 1st load processing is processing which loads the firmware corresponding to the hardware of an information processor to memory. Moreover, the 2nd load processing is processing which uses hardware resources through a firmware and loads an operating system to said memory. And secured processing is processing which secures the memory area of the size demanded by the 2nd load processing and the operating system on the above-mentioned memory.
[0053]
[Effect of the Invention] As explained above, according to this invention, a firmware is equipped with a flexible memory control unit, and the address dependence section which a firmware requires is minimized, and the mounting man day of the operating system which is a client is made to mitigate, and it becomes possible to mount more operating systems easily.
[0054] Moreover, flexible memory management becomes possible between an operating system and a firmware, and it is lost that an operating system is bound on the occasion of mounting in a

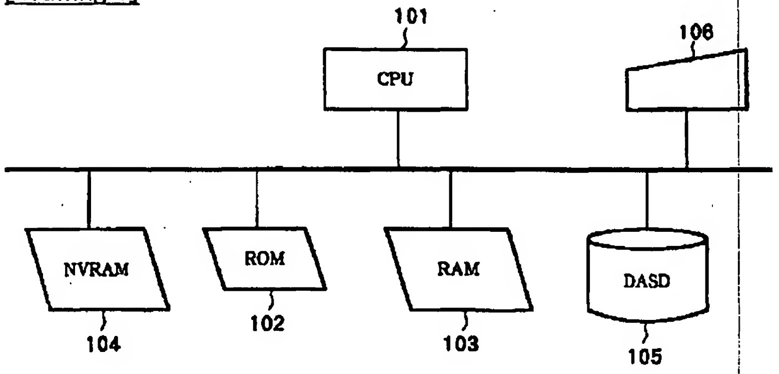
specific memory area.
[0055] Moreover, according to this invention, it becomes possible to secure from the address which had the memory area of size specified by means to load an operating system and this specified, and correspondence in various operating systems is attained.
[0056] Moreover, since the memory area which was equipped with the table showing the busy condition of room, and was demanded with reference to said table is secured according to this invention, grasp of the use situation of room and management become easy.
[0057] Moreover, according to this invention, since the above-mentioned table is automatically updated based on the secured condition of the memory area by the operating system etc., in subsequent starting processings, unless a system state is changed, suitable memory allocation is held, and the processing speed at the time of starting improves. That is, the overhead by memory relocation etc. can be minimized.
[0058] Moreover, since it becomes possible to edit the above-mentioned table according to this invention, more flexible memory management can be performed.
[0059] Moreover, since it becomes possible to move in the storing field of a firmware if needed on the occasion of reservation of the above-mentioned memory area according to this invention, the data in memory and flexible migration of a program are attained. For this reason, it becomes possible to reduce more the memory dependence which a firmware requires, and can respond to various operating systems.
[0060] Moreover, according to this invention, it realizes as one of the functions in which the firmware concerned is equipped with migration of the above-mentioned firmware.
[0061] Moreover, according to this invention, since the above-mentioned table is updated based on the result of migration of the above-mentioned firmware, a firmware will be loaded to a suitable location at the time of subsequent startings, unless a system is changed, migration of a firmware does not occur, but the processing effectiveness at the time of starting improves.
[0062]

[Translation done.]

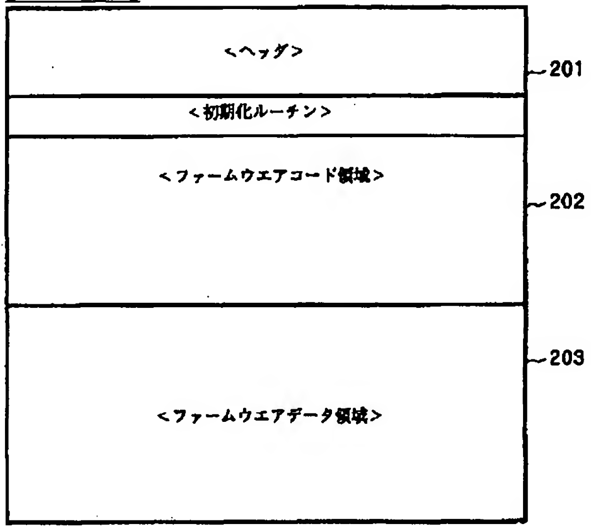
Ref. 1 (Drawings)

DRAWINGS

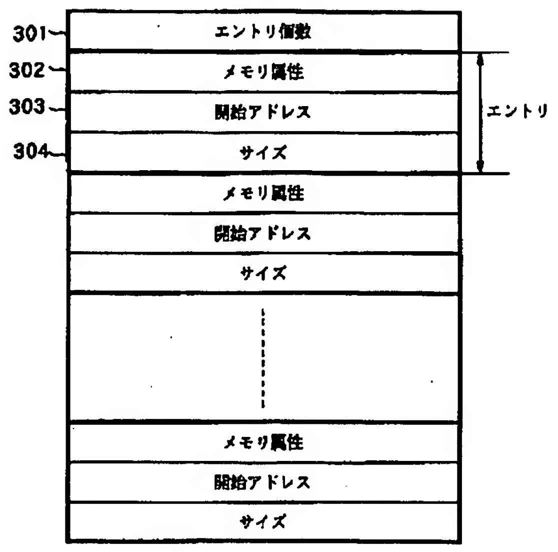
[Drawing 1]



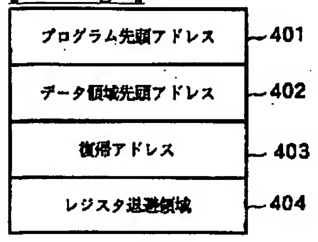
[Drawing 2]



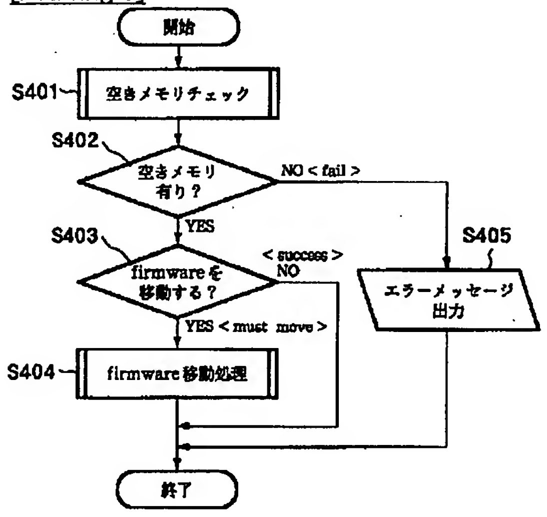
[Drawing 3]



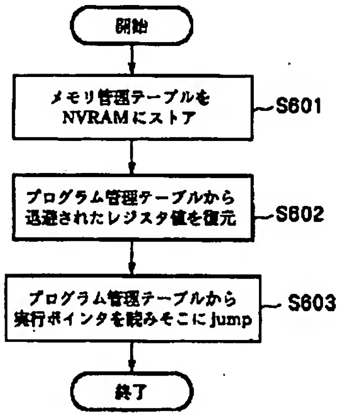
[Drawing 4]



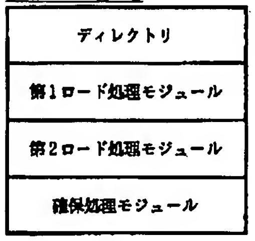
[Drawing 6]



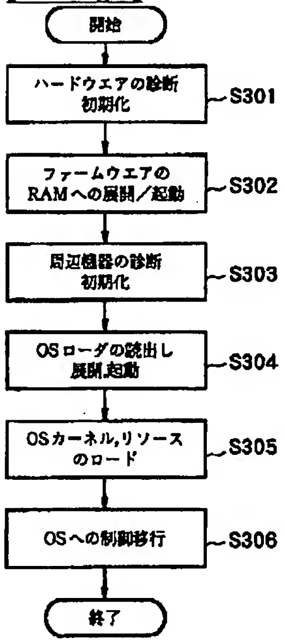
[Drawing 8]


開始
メモリ管理テーブルを NVRAMにストア
S601
プログラム管理テーブルから退避されたレジスタ値を復元
S602
プログラム管理テーブルから実行ポインタを読みそこにjump
S603
終了

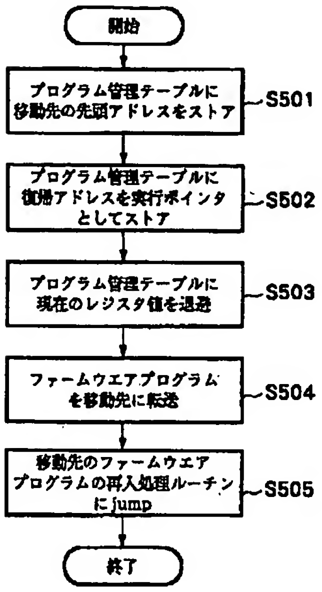
[Drawing 9]



[Drawing 5]



[Drawing 7]


開始
プログラム管理テーブルに移動先の先頭アドレスをストア
S501
プログラム管理テーブルに復帰アドレスを実行ポインタとしてストア
S502
プログラム管理テーブルに現在のレジスタ値を退避
S503
ファームウエアプログラムを移動先に転送
S504
移動先のファームウエアプログラムの再入処理ルーチンにjump
S505
終了

[Translation done.]